REMARKS

Claims 1-20 are pending in the present application. Claims 1, 2, 9, 10, and 20 are amended. No new matter is added by these amendments. Reconsideration of the claims is respectfully requested.

I. 35 U.S.C. § 103, Obviousness

The Office Action rejects claims 1-20 under 35 U.S.C. § 103(a) as being allegedly unpatentable over *Kumagai et al.* (U.S. Patent No. 6,108,714) in view of *Holland* (U.S. Patent No. 6,243,860). This rejection is respectfully traversed.

Kumagai teaches a method and apparatus for grouping programs that specifies a group of programs that is a combination of application programs that carry out a process by linking with each other. An operator of a data processing system may select two or more applications that are linked in some way as a group. An application window is generated for the group of applications. The applications are grouped at the data processing system so that the operator does not need to be conscious of the grouping functions. *Kumagai* also teaches that there is no need to create a parent application program. See *Kumagai*, col. 2, line 35, to col. 3, line 12.

Holland teaches a mechanism for exchanging information between a parent process and a child process compiled during execution of the parent process or between a run time compiler process and an application process. See *Holland*, Abstract.

The present invention provides a method, apparatus, and computer program product for chaining applications at a server. A client requests a chaining module and an option defines a service as a series of applications. Thus, a client computer may request multiple applications to be executed in series in order and the output of one application in the series may be passed to the input of the next application. Thus, the client computer need not invoke each application and pass results, possibly a significant amount of data, to the next application.

Kumagai does not teach or suggest receiving, at a chaining module, a request from a client computer for a service. The Office Action alleges that *Kumagai* teaches receiving a request for a service associated with a chaining module and a series of applications from an option corresponding to a chaining module at col. 2, lines 44-45;

col. 5, lines 47-51; col. 7, lines 31-35; and, col. 30, lines 1-14. The cited portions of *Kumagai* state:

> Still another object of the present invention is to provide an application program grouping method which specifies a group in a multi-window system, where the group is a combination of application programs which carry out a process by linking with each other, the multi-window system is capable of simultaneously executing in parallel a plurality of application programs respectively having one or a plurality of conversational or interactive windows that are displayed on a display, and the application program grouping method comprises the steps of (a) specifying at least two application programs which are to belong to one group, and (b) carrying out a grouping process which automatically generates one group made up of the specified application programs. According to the application program grouping method of the present invention, the application programs can be linked by a simple operation, and the operation ease is improved. Because the operator does not need to be conscious of the grouping functions, the load on the operator is reduced, thereby making it possible to improve the operation efficiency. Furthermore, there is no need to create a parent application program as in the case of the first conceivable application grouping method described above.

Kumagai, col. 2, lines 35-55.

> In this embodiment, application programs (hereinafter simply referred to as "applications) A and B are linked using a common conversational or interactive window system structuring tool 2 as shown in FIG. 3. Each of the applications A and B is made up of a main application body 1 and the structuring tool 2. The structuring tool 2 includes a generating function which generates an application window 3, a communicating function which communicates between the applications A and B, and a describing function which describes a grouping of applications using the communicating function. More particularly, the describing function in this embodiment is a function of changing a display format, such as changing the background color within the application window 3 and changing a frame color of the application window 3. When changing the display format, the method of changing the frame color of the application window 3 is particularly preferable in that a data processing

> quantity associated with the display format change is small. The main application body 1 generates the application window 3 immediately after the application is started by use of the structuring tool 2. The main application body 1 itself makes a transmission process when a link with another application becomes necessary, and makes a reception process when a link is requested from another application.
>
> In the following description, it will be assumed for the sake of convenience that the plurality of applications which are simultaneously executable in parallel respectively have 1 window. However, each application may of course have a plurality of windows.

Kumagai, col. 5, lines 25-53.

> When the pairing shown in FIG. 7B is made, the system manager SM makes a message communication to notify the application A that the application C is the linked application forming the pair with the application A, as shown in FIG. 9. In addition, the system manager SM makes the message communication to notify the application C that the application A is the linked application forming the pair with the application C.

Kumagai, col. 7, lines 31-38.

> 24. The application program grouping apparatus as claimed in claim 23, wherein the automatic grouping request is selected from a group consisting of:
> a new group automatic grouping request which requests automatic grouping of each application program started after generation of the automatic grouping request as a new group;
> a group adding type automatic grouping request which requests automatic grouping such that each application program started after generation of the automatic grouping request is added to an existing group; and
> a group reserve type automatic grouping request which requests automatic grouping of each application program started after generation of the automatic grouping request if each started application is included in a reserved group.

Kumagai, col. 29, line 12, to col. 30, line 13. Nowhere in the cited portion or any other portion does *Kumagai* teach or suggest receiving a request for a service associated with a

chaining module and a series of applications from an option. More particularly, *Kumagai* does not teach or suggest receiving a request from a client for a service, wherein an option identifies the service as a series of applications, as recited in claim 1, for example. *Kumagai* is concerned with grouping applications in a data processing system for ease of operation without creating a parent application. *Kumagai* does not teach or suggest a receiving, at a chaining module, a request from a client computer.

As acknowledged in the Office Action, *Kumagai* does not teach or suggest executing the series of applications in order and passing the output of each application to the input of the next application in the series. In fact, *Kumagai* teaches away from a series of applications, because *Kumagai* teaches that the applications to be linked are simultaneously exccutable in **parallel**. See col. 5, lines 25-53, shown above.

To make up for the deficiencies of *Kumagai*, the Office Action alleges that *Holland* teaches executing a series of applications in order and passing the output of each application to the input of the next application in the series at FIG. 2A; col. 5, lines 60-67; and col. 20, lines 1-20. FIG. 2A of *Holland* is as follows:

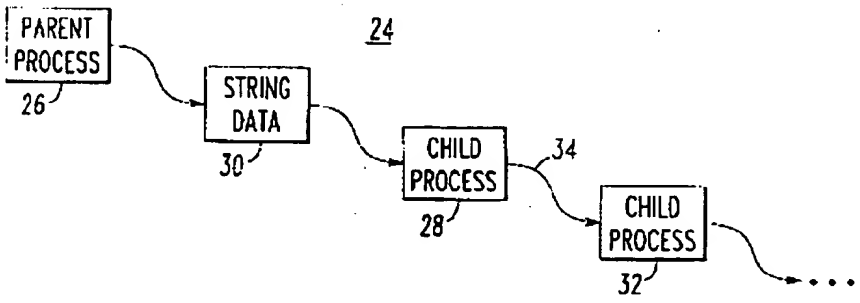

FIG. 2A
PRIOR ART

The cited portions of the description state:

> Referring to FIGS. 2A and 2B, dynamic shell objects illustrate the difference between a UNIX shell 24 and the processor shell 16 for the compiler 12 of FIG. 1. In the UNIX shell 24 of FIG. 2A, string information 30 flows from (but not to) a parent process 26 to (but not from) a child process 28. Also, string information 34 flows from (but not to) the child process 28 to (but not from) another child process 32.

Holland, col. 5, lines 60-67.

> A parent process can dynamically create a child from which it can receive information, and interact normally.
>
> With the compiler 12, the data atom wrapper allows for an inherent sharing of memory without employing complex and expensive programming techniques such as data-piping. The compiler 12 may be employed to improve computer process automation. The compiler language introduces a unique POPS technique that compliments strongly typed OOP with what is believed to be novel object processing. The compiler 12 automates verification of customizable process components which has broad application for engineering in highly regulated industries (e.g., the nuclear industry). Other language features, such as procedural objects, object inter-process communication, and variable input/output lists significantly advance the automation of engineering analyses (e.g., automating nuclear reactor design safety analysis or the design of nuclear reactor cores).

Holland, col. 20, lines 1-20. Thus, *Holland* merely teaches that string information flows from a parent process to a child process and that a data atom wrapper allows for inherent sharing of memory without employing data-piping. However, *Holland* does not teach or fairly suggest executing a series of applications that make up a requested service and passing the output of one application to the input of the next application in the service. More particularly, *Holland* does not teach executing a series of applications in order in the context of a requested service at a chaining module at a server, as recited in claim 1, for example.

Moreover, the Office Action may not use the claimed invention as an "instruction manual" or "template" to piece together the teachings of the prior art so that the invention is rendered obvious. *In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992). Such reliance is an impermissible use of hindsight with the benefit of Applicant's disclosure. *Id.* Therefore, absent some teaching, suggestion, or incentive in the prior art, *Kumagai* and *Holland* cannot be properly combined to form the claimed invention. As a result, absent any teaching, suggestion, or incentive from the prior art to make the proposed combination, the presently claimed invention can be reached only through an impermissible use of hindsight with the benefit of Applicant's disclosure a model for the needed changes.

Kumagai and *Holland*, taken individually or in combination, fail to teach or suggest each and every claim limitation. Therefore, the proposed combination of *Kumagai* and *Holland* do not render independent claim 1 obvious. Independent claims 9, 12, and 20 recite subject matter addressed above with respect to claim 1 and are allowable for similar reasons. Since claims 2-8, 10, 11, and 13-19 depend from claims 1, 9, and 12, the same distinctions between *Kumagai* and *Holland* and the invention recited in claims 1, 9, and 12 apply for these claims. Additionally, claims 2-8, 10, 11, and 13-19 recite further limitations not taught or suggested by the references.

More particularly, claims 2 and 10 recite that the option is specified in a properties file and associates the service with a name. The Office Action alleges that *Kumagai* teaches these features at col. 7, lines 2-20, which is shown above, and col. 1, lines 50-56, which reads as follows:

> On the other hand, according to a second conceivable application grouping method, the applications to be linked can be started independently, and the application to be linked to another is specified by an application name. FIG.2 shows a case where the application A can start the applications B having the same application name by specifying the application name.

Neither the cited portion nor any other portion of *Kumagai* makes any mention of a properties file. Applicants submit that neither *Kumagai* nor *Holland* teaches or suggests a properties file in which a service is registered and associated with a name.

With respect to claim 6, for example, the Office Action alleges that *Holland* teaches that a first application within the series of applications is a verification application at col. 20, lines 10-12, which is shown above. Clearly, the cited portion states that the **compiler** automates verification. *Holland* does not teach or suggest that an application within a series of applications that form a service requested by a client computer is a verification application.

Furthermore, with respect to claims 7 and 8, for example, the Office Action alleges that *Holland* teaches that a first application and a second application within the series of applications are language translation engines at col. 1, lines 14-20, and col. 19, lines 25-32. The cited portions of *Holland* state:

> A compiler includes a computer process or program which translates a computer language into machine code. An intermediate compiler includes a compiler which translates a higher level computer language into a lower level computer language which is not yet optimized into true machine code.

Holland, col. 1, lines 14-20.

> Data pedigree tracking capability could be somewhat mimicked with other interpreters, such as Perl and Tcl/Tk, to a very limited extent. However, other interpreters require a data pipe to move information back and forth between the interpreter and the application. Such a process is computationally expensive compared to the exemplary compiler 12 which shares memory and compiles user input during application run time.

Holland, col. 19, lines 25-32. While the cited portions do nominally mention translating programs from one computer language to another and interpreting other computer languages, there is no mention in either of the applied references of a series of applications identified as a service to be executed in order, wherein one or more of the applications is a language translation application.

Therefore, Applicant respectfully requests withdrawal of the rejection of claims 1-20 under 35 U.S.C. § 103(a).

II. Conclusion

It is respectfully urged that the subject application is patentable over the prior art of record and is now in condition for allowance.

The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: March 15, 2005

Respectfully submitted,

Stephen R. Tkacs
Reg. No. 46,430
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Agent for Applicant